

DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

Prof. Delano M. Beder (UFSCar)

Atividade AA-3: Sistema para agendamento de consultas online com profissionais

Obs 1: Essa atividade deve ser baseada na atividade AA-2. Ou seja, deve-se apenas implementar os novos requisitos (funcionalidades providas em uma REST API) aqui mencionados -- levando em consideração o que já foi desenvolvido na atividade AA-2.

O sistema deve incorporar os seguintes requisitos.

- REST API -- CRUD ¹ de clientes
 - Cria um novo cliente [Create - **CRUD**]
POST <http://localhost:8080/clientes>
Body: raw/JSON (application/json)
 - Retorna a lista de clientes [Read - **CRUD**]
GET <http://localhost:8080/clientes>
 - Retorna o cliente de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/clientes/{id}>
 - Atualiza o cliente de id = {id} [Update - **CRUD**]
PUT <http://localhost:8080/clientes/{id}>
Body: raw/JSON (application/json)
 - Remove o cliente de id = {id} [Delete - **CRUD**]
DELETE <http://localhost:8080/clientes/{id}>
- REST API -- CRUD de profissionais
 - Cria um novo profissional [Create - **CRUD**]
POST <http://localhost:8080/profissionais>
Body: raw/JSON (application/json)
 - Retorna a lista de profissionais [Read - **CRUD**]
GET <http://localhost:8080/profissionais>
 - Retorna o profissional de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/profissionais/{id}>
 - Retorna a lista de todos os profissionais de especialidade cujo nome = {nome}
GET <http://localhost:8080/profissionais/especialidades/{nome}>
 - Atualiza o profissional de id = {id} [Update - **CRUD**]
PUT <http://localhost:8080/profissionais/{id}>
Body: raw/JSON (application/json)
 - Remove o profissional de id = {id} [Delete - **CRUD**]
DELETE <http://localhost:8080/profissionais/{id}>

- REST API -- Retorna a lista de consultas [Read - **CRUD**]
GET <http://localhost:8080/consultas>
- REST API -- Retorna a consulta de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/consultas/{id}>
- REST API -- Retorna a lista das consultas do cliente de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/consultas/clientes/{id}>
- REST API -- Retorna a lista das consultas do profissional de id = {id} [Read - **CRUD**]
GET <http://localhost:8080/consultas/profissionais/{id}>

Obs 2: Em todas as funcionalidades mencionadas acima, não há necessidade de autenticação (login)

Dica: Na configuração do *Spring Security* utilize algo semelhante ao apresentado no código abaixo:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable().authorizeRequests()
        // Controladores REST
        .antMatchers("/clientes", "/profissionais", "/consultas").permitAll()
        .antMatchers("/clientes/{\\d+}", "/profissionais/{\\d+}").permitAll()
        .antMatchers("/consultas/{\\d+}").permitAll()
        .antMatchers("/profissionais/especialidades/{\\w+}").permitAll()
        .antMatchers("/consultas/clientes/{\\d+}").permitAll()
        .antMatchers("/consultas/profissionais/{\\d+}").permitAll()
        // Demais linhas
        .anyRequest().authenticated()
        .and()
        .formLogin().loginPage("/login").permitAll()
        .and()
        .logout().logoutSuccessUrl("/").permitAll();
}
```

Arquitetura: Modelo-Visão-Controlador

Tecnologias

- Spring MVC (Controladores REST), Spring Data JPA, Spring Security & Thymeleaf (Lado Servidor)

Ambiente de Desenvolvimento

- A compilação e o *deployment* deve ser obrigatoriamente ser realizado via *maven*.
- Os arquivos fonte do sistema devem estar hospedados obrigatoriamente em um repositório (preferencialmente github).